Comparing the Energy Efficiency of CMP and SMT Architectures for Multimedia Workloads *

Ruchira Sasanka, Sarita V. Adve, Yen-Kuang Chen†, and Eric Debes†

University of Illinois at Urbana-Champaign
Department of Computer Science
{sasanka,sadve}@cs.uiuc.edu,

†Intel Corporation
Microprocessor Research Labs
{yen-kuang.chen,eric.debes}@intel.com

UIUC CS Technical Report UIUCDCS-R-2003-2325, March 2003

Abstract

Chip multiprocessing (CMP) and simultaneous multithreading (SMT) are two recently adopted techniques for improving the throughput of general-purpose processors by using multithreading. These techniques are likely to benefit the increasingly important real-time multimedia workloads, which are inherently multithreaded. These workloads, however, often run in an energy constrained environment. This paper compares the energy efficiency of CMP and SMT for multimedia applications.

Assuming out-of-order processors as the core components, we investigate the design space by varying the core processor complexity and using a range of frequencies. To measure energy efficiency, we compare the energy consumed by systems that provide the same performance in the entire design space. We find that across the performance spectrum, a CMP configuration is the most energy efficient for our systems and applications. Further, CMP processors are amenable to further energy reductions through the use of recently proposed adaptive techniques.

Finally, since SMT can provide benefits for single-thread performance, we propose a hybrid CMP/SMT architecture consisting of a CMP with SMT processor cores. This architecture shows significantly better energy-efficiency than pure SMT, and is a good compromise solution that achieves both the high single-thread performance of SMT and the energy efficiency of multithreaded CMP.

*This work is supported in part by the National Science Foundation under Grant No. CCR-0096126, EIA-0103645, CCR-0209198, and CCR-0205638, a gift from Motorola Inc., and the University of Illinois. Sarita V. Adve is also supported by an Alfred P. Sloan Research Fellowship. Part of the work was done as a summer internship project at Intel MRL.

1 Introduction

General-purpose processors have begun to support multithreading for improved throughput, using either chip multiprocessors (CMP) [8] or simultaneous multithreading (SMT) [18]. Multithreading support is potentially a good match for multimedia applications which are inherently multithreaded, and are becoming an increasingly important workload for general-purpose processors. These applications, however, often run in an energy-constrained environment, and energy efficiency is an important criterion for evaluating architectures suitable for them. This paper evaluates the energy efficiency of general-purpose CMP and SMT architectures for multimedia applications.

A fair comparison of the energy efficiency of CMP and SMT must consider the design space of the possible alternative configurations for these architectures and the performance of the configurations that are compared. This work focuses on out-of-order superscalar processors, based on contemporary general-purpose designs. To ensure a fair comparison, we consider a range of possible core processor complexities, ranging from 2-wide to 8-wide fetch bandwidth. To consider a full performance continuum, we evaluate the considered processor core architectures over a range of frequencies (from 100 MHz to 1 GHz). The configurations of a core architecture at different frequencies may be interpreted as a single processor supporting dynamic frequency and voltage scaling (DVS) or as different fixed-frequency processor designs. We compare the energy efficiency of CMP and SMT by considering configurations that provide the same performance for a given workload, and perform such comparisons for all performance points in the investigated design space. We consider two-thread and four-thread workloads derived from combinations of eight (sequential) multimedia benchmarks consisting of low and high bit rate video and speech codecs.

We find that across the entire performance spectrum studied, CMP generally provides the most energy efficient configuration for our systems and applications. The difference between the least-energy CMP and the least-energy SMT configurations is generally higher in the higher performance regions, and is generally higher for workloads with high total IPC. In general, the least-energy CMP configuration has a moderately lower complexity processor core than the least-energy SMT configuration at any given performance point. This is because the SMT must exploit its performance from one processor core while the CMP has multiple such processor cores available. Thus, the CMP has higher total resources than the SMT; however, it uses these resources in a more energy efficient way than the SMT.

We show that the factors responsible for the above results include (1) the relatively steep slope of the power vs. complexity curve in modern out-of-order processors, which makes the higher complexity SMT processor core significantly more expensive than the CMP processor core, and (2) the use of aggressive clock gating, which reduces the energy cost of the larger number of under-utilized resources in the CMP. We also identify situations where SMT may do better than CMP; however, we did not see these situations in our workloads.

Although CMP configurations give the best energy efficiency, different configurations are optimal at different performance points (as is the case with SMT). This motivates the use of recently proposed adaptive architecture and frequency/voltage scaling techniques. Further, we find that applying these techniques independently on the different CMP processor cores to create a heterogeneous CMP provides even further energy savings for CMP.

Our results clearly underscore the advantage of CMP for the systems and multithreaded workloads studied. Nevertheless, wider SMT processors can be more efficient for single thread performance for a large class of workloads. With this in mind, we propose a new hybrid CMP/SMT architecture where a CMP is built out of SMT cores. We find such a two processor CMP with two-thread SMT cores has significantly higher energy efficiency than a pure SMT processor. Such a hybrid architecture may therefore provide a good compromise solution to get both the high single-thread performance of SMT and the high multithreaded energy efficiency of CMP. Furthermore, for current SMT processors supporting two threads, this hybrid architecture provides a good migratory path to a CMP.

Although there is significant prior work comparing the performance of CMP and SMT [6, 8] and comparing the energy efficiency of SMT with a superscalar [16], there is very little prior work on energy related comparisons of SMT and CMP. The only such work, to our knowledge, is by Kaxiras et al. and also in the context of multimedia applications [14]. However, that work considers a VLIW processor core (which produces very low IPC for the compiled codes studied) and compares average power at a given frequency for CMP vs. SMT. Further, it examines only two alternative core architectures and only one workload. In contrast, we study out-of-order superscalar processors (which give higher IPCs) and compare energy at the same performance, for a wider range of workloads and core processor architectures. That work concludes that SMT consumes more power than CMP (at a given frequency). This does not contradict our work since we also find that all CMP configurations have higher power than all SMT configurations at a fixed (highest) frequency (e.g., Figure 5 in Section 3.1.2). Our main results find CMP to be superior based on a comparison of energy at equal performance, where CMP often runs at a lower frequency than SMT.

Variable Processor Parameters	
Fetch/decode/retire rate	$Width \in \{2, 3, 4, 5, 6, 8\}$
Instruction window (reorder buffer) size	$Width$ * 16
# of Integer functional units	$Width$
# of Floating point units	1 if $Width \leq 4$, else 2
Integer register file size	32*N + reorder buffer size
Float register file size	32*N + reorder buffer size
Common Processor Parameters	
Processor speed	100 MHz to 1 GHz (voltage scaled)
Address generators	2
Integer FU latencies	1/4/12 add/multiply/divide (pipelined)
FP FU latencies	4 default, 12 div. (all but div. pipelined)
Memory queue size	32 entries
Branch prediction	2KB bimodal agree, 32 entry RAS
Common Memory Hierarchy Parameters	
L1 data cache	16KB, 4-way associative,
(one per processor core)	64B line, 2 ports
L2 cache (one per system)	4MB, 8-way associative,
	64B line, 1 port
Main Memory	16B/cycle, 4-way interleaved
Common Contentionless Memory Latencies	
L1 (data) hit time	2 cycles
L2 hit time	12 cycles
Main memory	100 cycles

Table 1. Processor and memory parameters.

2 Experimental Methodology

2.1 Systems Modeled

We model SMT processors with support for two or four threads; CMP systems with two or four single-thread processors (also referred to as two-thread and four-thread CMP respectively); and a combined CMP/SMT system consisting of two two-thread SMT processors. In all cases, the

processor core is an out-of-order superscalar, similar to the MIPS R10000. To adequately represent the design space, we model several configurations for the processor core, ranging from a fetch/retire width of two to eight. For each fetch/retire width, we appropriately scale the other resources (e.g., instruction window size and the number of functional units). Table 1 summarizes the various processor core parameters – *Width* refers to the fetch/retire width and *N* is the number of threads supported by the processor core. For CMP, our main results assume that all processor cores have the same configuration. We denote a CMP or SMT system with processors with a fetch/retire width of N by *CMP-N* and *SMT-N* respectively.

For SMT, we assume that all concurrent threads share most resources in the processor, including the instruction window, functional units, and register files; however, each thread is given a separate branch prediction table. Further, 32 additional integer and floating point registers are assumed for each additional thread, to capture the architectural state. We use the ICOUNT policy [18] to prioritize instruction fetch from different threads.

We assume a two-level data cache hierarchy and a perfect instruction cache. For CMP, each processor has its own L1 cache, and these caches are connected to a common L2 cache through a bus. The memory hierarchy parameters are also summarized in Table 1. In particular, each processor core has a 16K L1 data cache for both CMP and SMT. This size supports all the four-thread workloads studied (Section 2.2), and further increase in size does not appreciably improve performance. Note that a CMP configuration with N processors has N times the L1 data cache size as an SMT, which may appear to give an unfair advantage to SMT. In real systems, it is likely that the total cache for an N processor CMP will be somewhat larger than that for an N-thread SMT, but probably not N times larger. Since our results showed that the energy efficiency of an SMT was much lower than that of a CMP, we chose to report results with system parameters that would give the best showing to SMT. Overall, we found that neither the energy nor the performance results are very sensitive to the L1 cache size, since these applications exhibit very high hit rates and the L1 cache is not the dominant consumer of energy.[1] For the same reason, we found that two ports for the L1 cache were sufficient, even for SMT.

Finally, we consider processor frequencies in the range of 100 MHz to 1 GHz. For energy computation, we derive voltages corresponding to these frequencies using the formula $f = k\frac{(V-V_{th})^2}{V}$ where f is the frequency, V is the supply voltage, V_{th} is the threshold voltage, and k is a technology parameter. We deduced values for k and V_{th} by fitting published frequency-voltage pairs for the Intel XScale processor (which supports dynamic voltage and frequency scaling) in the above equation [12] as done in [9].

[1] One could argue that a real general-purpose system will have much more cache than 16KB per processor to adequately handle other workload domains. Our results will hold for those systems, assuming support to deactivate the unused portion of the cache [1].

App.	Type	Input Size (Frames)	Base IPC
GSMdec	Speech	1000	3.4
GSMenc	codec	1000	3.7
G728dec	Speech	1000	2.1
G728enc	codec	1000	1.8
H263dec	Video	450	3.3
H263enc	codec	50	1.6
MPGdec	Video	200	2.6
MPGenc	codec	50	1.4

Table 2. Benchmarks.

2.2 Workloads

We consider eight (sequential) multimedia benchmarks covering high and low bit rate video and speech codecs. These benchmarks are summarized in Table 2 and described in more detail in previous work [9]. They form the core components of many high-level multimedia applications; e.g., video teleconferencing. A real system would run several of these benchmarks together as part of one or more such high-level applications. For example, a video teleconferencing application between N participants at N different sites would involve one video and speech encoder and N-1 video and speech decoders at each site. A participant may receive high or low bit rate streams depending on the computation power and bandwidth available at the other participating sites; therefore, a site may need to support different types of decoders and encoders within the same application. In general, we can envisage realistic workloads consisting of a number of different copies of different combinations of the applications in Table 2.

For the small-scale systems studied here (two or four thread CMP and SMT), we can assume that the total number of threads available for running in a realistic system will be larger than the number of simultaneous threads supported in the system. A real-time operating system (RTOS) must therefore choose which combination of threads to co-schedule at each time. The co-scheduling algorithm can have an impact on the overall performance, but real-time co-scheduling algorithms for SMT are still an active area of research [13]. To eliminate dependence on the co-scheduling algorithm, we report results separately for different combinations of N threads for an N-thread system. Thus, for a two-thread system, we separately report results for different pairs of the eight benchmarks of Table 4. The actual performance of a full real-time application would depend on the frequency with which the RTOS co-schedules each of the

specific combinations for its chosen co-scheduling policy.

Furthermore, a real RTOS co-scheduling policy may co-schedule different parts of N concurrent benchmarks at different times. For example, consider two benchmarks with very different execution times per frame. The shorter frame may be co-scheduled along with any part of a longer frame and it is possible the execution characteristics are different depending on when the two frames are co-scheduled. Again, to report results independent of the co-scheduling policy and to average out such differences, we run several frames of the co-scheduled benchmarks without any synchronization across frame boundaries, to get the average behavior for that benchmark combination. The maximum number of frames we consider for each benchmark is reported in Table 2. For each combination of benchmarks, we run until the shortest one completes. The number of frames that the longer benchmarks complete within this time could vary depending on the architecture configuration. We partly alleviate the impact of this variation by reporting energy normalized to the total number of instructions executed. The alternative method of fixing the number of frames of the longer benchmarks would mean that for some runs, we would only be running a single thread for part of the time (and this time would vary depending on the processor core architecture). This would most likely be unfair to the energy efficiency of an SMT system since we assume that the RTOS always has another thread that is ready to run.[2]

Studying all combinations of two and four out of our eight benchmarks would have resulted in an inordinately large number of workloads (e.g., 36 possibilities for 2-thread systems). We therefore selected a subset of these, summarized in Table 3, using the following methodology. We determined that our results were sensitive to the total IPC. A higher symbiosis means the SMT is well utilized. for the combination. For the two-thread case, we therefore divided all the possible benchmark pairs into four categories, based on the sum of the IPCs of the two benchmarks when run individually on the most aggressive processor (described in Section 2.1). From each category, we then selected at least two workloads to represent the range of symbiosis[3] values in that category. For the four-thread case, we considered all possible pairs of the two-thread workloads and used the same criteria as for the two thread workloads. For the four thread CMP/SMT architecture, we used the four-thread workloads – the constituent two-thread pairs were paired again for each SMT processor.

[2]For the CMP system, we could just as well have chosen to run a fixed amount of work and then deactivated the CMP processors that finished their work ahead of the others.

[3]Symbiosis is defined as *effectiveness with which multiple jobs achieve speedup when run on multithreaded machines* [17].

2-thread workload	Total IPC
MPGenc_MPGenc	2.6
MPGenc_G728dec	3.5
MPGenc_MPGdec	3.9
G728enc_G728dec	3.9
H263enc_MPGdec	4.1
MPGenc_GSMdec	4.7
H263enc_H263dec	4.8
H263dec_G728enc	5.0
H263enc_GSMenc	5.2
H263dec_G728dec	5.4
GSMenc_G728dec	5.8
MPGdec_GSMdec	5.9
H263dec_GSMenc	6.9
GSMenc_GSMenc	7.3

(a)

4-thread workload	Total IPC
MPGenc_MPGenc_MPGenc_MPGenc	5.3
MPGenc_MPGenc_G728dec_G728enc	6.6
H263dec_H263enc_GSMenc_H263enc	10.0
H263dec_H263enc_G728dec_H263dec	10.2
MPGdec_MPGenc_GSMenc_GSMenc	11.2
G728dec_H263dec_GSMdec_MPGdec	11.3
GSMenc_H263dec_GSMenc_H263dec	13.8
GSMenc_GSMenc_GSMenc_GSMenc	14.6

(b)

Table 3. (a) Two-thread and (b) four-thread workloads. Each set is ordered by the sum of the IPCs of the constituent threads when run individually on the most aggressive processor configuration.

2.3 Simulation Environment and Methodology

We model the performance of the systems in Section 2.1 using a version of the RSIM simulator [10] modified to support both SMT and CMP. Previous work showed that for the benchmarks studied here, performance scales with processor frequency, since the amount of time spent on memory stalls is negligible [9]. We therefore run simulations at one base frequency to obtain the number of execution cycles, and appropriately scale this number to obtain execution time at different frequencies.

To model energy, we use the Wattch tool [4] integrated with RSIM. We use clock gating as implemented in Wattch so that only 10% of the maximum power is charged for a resource that is not used in a given cycle. We also model the energy consumption of the bus between the L1 and L2 caches in the CMP, using models from the Orion project [19]. [4] We conservatively assume a 5mm bus length with two processors and a 10mm bus length with 4 processors. We do not report energy for the L2 cache here since it can be placed either on-chip or off-chip, and the optimal size of the L2 cache can vary depending on the number of threads and the workloads.

Henceforth, when we refer to a set of *core architec-*

[4]We thank Li-Shiuan Peh and Hang-Sheng Wang for quickly generating and providing us the bus models.

tures, we refer to the set of processors with variations in fetch/retire width. When we refer to a set of *configurations*, we refer to combinations of the core architecture and frequency (in the context of a two-thread CMP or SMT).

3 Results

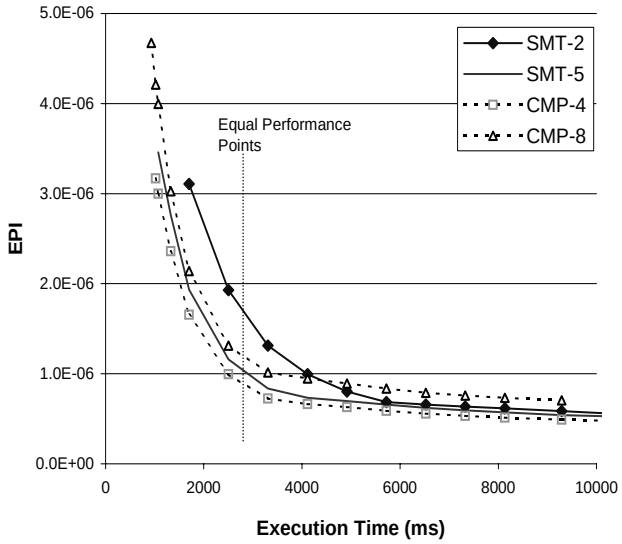

Figure 1. Example EPI vs. execution time graph.

When considering energy as a metric of comparison, one must also consider the performance obtained. One common metric used is the energy-delay product (i.e., energy divided by performance) [7]. However, this metric is unsatisfactory if the user desires a fixed amount of performance or is constrained by a fixed amount of energy (e.g., battery life). Specifically for real-time applications such as studied here, there is often a fixed desirable performance target (derived from the application deadlines and the rest of the load on the system). We therefore focus most of this section on understanding optimal energy configurations, given a fixed performance target (Section 3.1). The data and analysis for determining the optimal performance configuration for a fixed energy target is similar (see below), and we do not present that here for lack of space. We do show the results for energy-delay product and other more conventional metrics in Section 3.2.

3.1 Energy Optimality for a Performance Target

3.1.1 Representing the Data

For an SMT or CMP with a given core architecture, varying the processor frequency provides a continuum of performance points. Any of these points could be designed either as a fixed-frequency design, or could be invoked in a system with DVS support. We collect data for SMT and CMP systems using all combinations of core architectures and frequencies given in Section 2.1. For each workload, we then compare points with equal performance among all system configurations, to determine which system gives the least energy for that performance. The optimal system could be different for different performance points.

Figure 1 illustrates the above. It plots energy per instruction (or EPI) versus execution time for the MP-Genc_MPGdec workload. Each curve in the figure represents one core architecture for an SMT or CMP system. This figure shows SMT-2, SMT-5, CMP-4, and CMP-8. Each point on a core architecture curve represents a different frequency. The points along a vertical line on this graph represent points of equal performance. The lowest point on the line represents the configuration that gives the least energy for that performance. For example, for a target execution time of 2,500ms, CMP-4 gives the least-energy. This architecture also turns out to be the least-energy one for a large performance range for this application.[5]

In general, the least-energy architecture may be different in different performance ranges for two reasons. First, not all architectures may be able to provide all performance points in the extreme right and left sides of the execution time axis. For example, in Figure 1, CMP-8 is least-energy for a few points in the left most execution time, because CMP-4 is not able to provide that high a performance even at the maximum frequency. Second, in the middle performance ranges, the least-energy configuration could change if the curves of two core architectures cross. This crossing can occur when at least one of the core architectures is operating at a relatively low voltage, where frequency is not linearly proportional to voltage.

To distill the information from the EPI-execution time graphs into a more readable form, we divide the execution time axis into intervals. We start a new interval at a performance point where the least-energy SMT or CMP core architecture changes or when the least-energy overall architecture changes. We report EPIs for one performance point in each such interval – we choose the point where the difference between the least-energy SMT and least-energy CMP architecture is the maximum. Figure 2 shows this data for two-thread workloads running on two-thread SMT and CMP configurations. For lack of space, we show graphs for only 8 workloads. At each of the above performance points, the figure shows the EPI for the least-energy CMP and SMT architecture, and gives the fetch/retire width of the architecture below the corresponding bar. Figure 3 shows analogous data for four-thread workloads (for four-thread CMP and SMT). In addition, at each performance point, the figure also shows the EPI for the combined CMP-SMT system.

[5]The representation and analysis for understanding the highest-performance configurations with a fixed energy constraint is analogous. A horizontal line on the EPI-execution time graph identifies the equal-energy points, and the leftmost point identifies the highest-performance configuration.

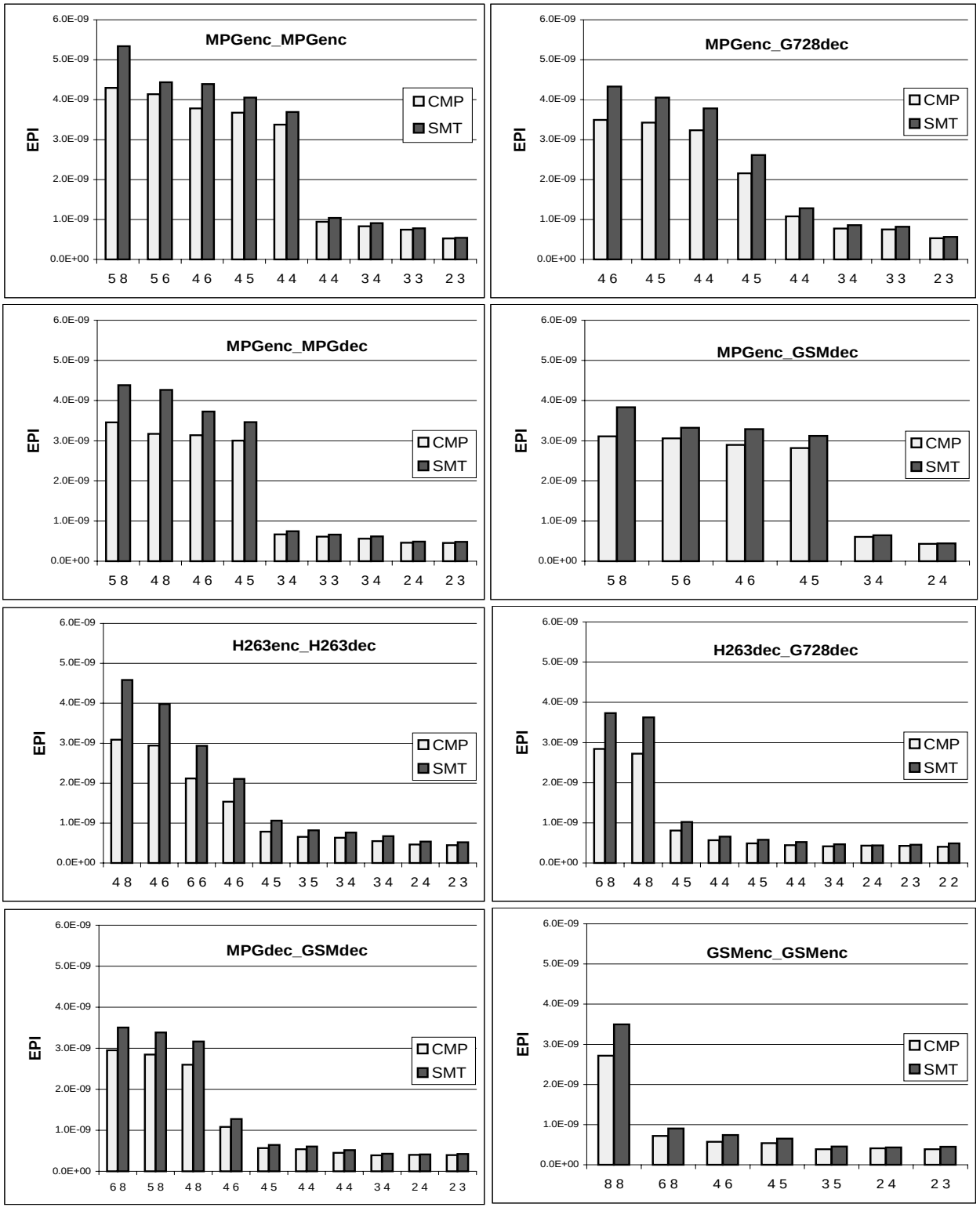

Figure 2. **EPI for best-energy CMP and best-energy SMT configuration at different performance points for two-thread workloads.The number under the bar gives the fetch/retire width of the core in that configuration.**

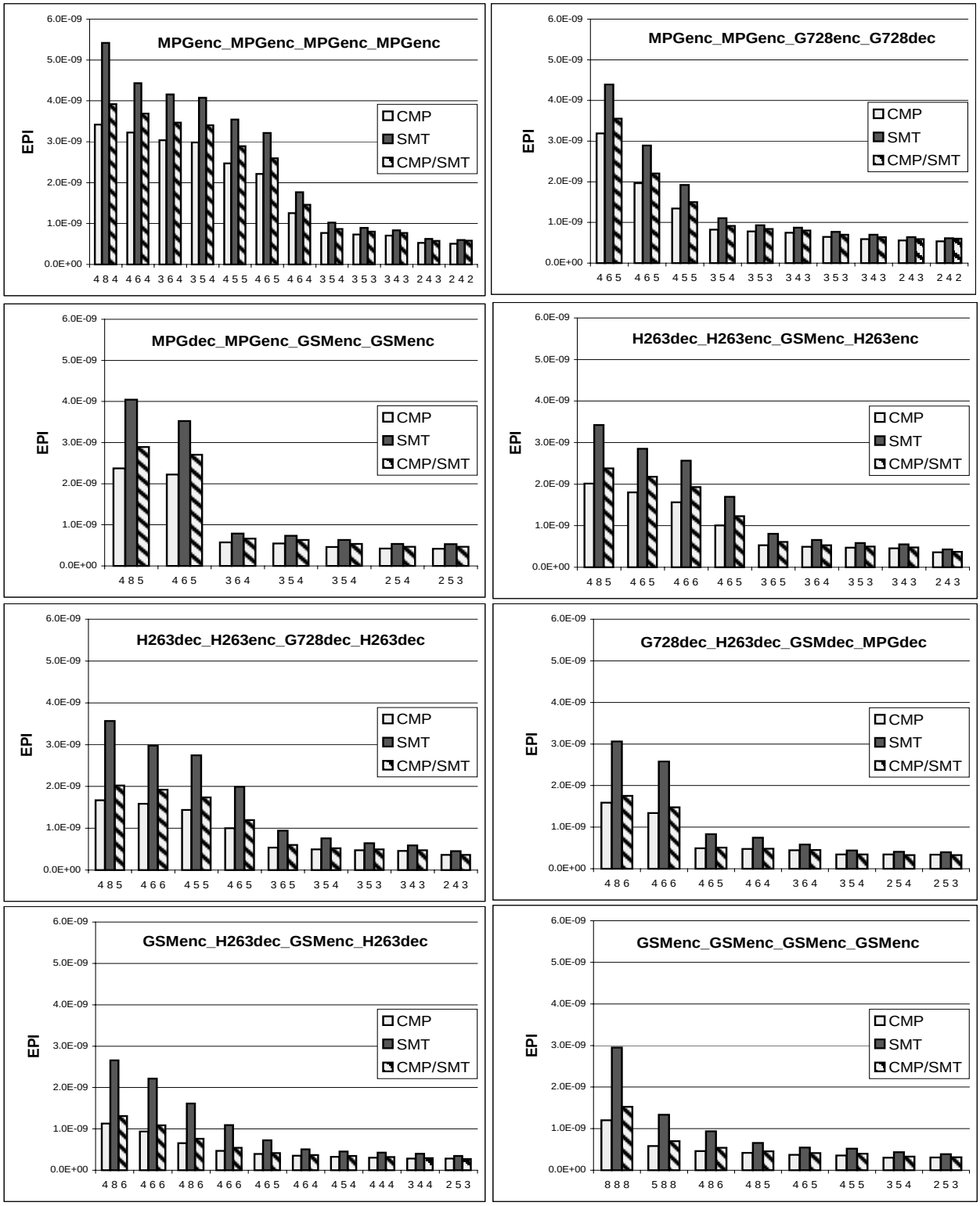

Figure 3. **EPI for best-energy CMP, best-energy SMT and the best-energy CMP/SMT configuration at different performance points for four-thread workloads.The number under the bar gives the fetch/retire width of the core in that configuration.**

Tables 4(a) and (b) supplement the above graphs by tabulating the magnitude of the EPI difference between the best SMT and CMP configurations (as a percentage of the SMT configuration). These tables divide the execution time axis on the EPI-execution time graphs into three regions (A, B, and C), based on the performance degradation relative to the highest performance configuration (which is always CMP-8 for all workloads). The performance degradation is from 1X to 2.5X for region A, 2.5X to 6X for region B, and $>$ 6X for region C. For each region, the tables give the range of the percentage improvement (in EPI) of the best CMP configuration over the best SMT configuration at the different performance points in this region. (We need to give a range because the improvement at different points is different.) Note that for a given region in this table, the optimal CMP and SMT architectures may be different at different points in the region.

Workload	IPC	A	B	C
MPGenc_MPGenc	2.6	6-19	3-4	1-5
MPGenc_G728dec	3.5	14-19	6-15	5-9
MPGenc_MPGdec	3.9	12-25	7-13	5-8
G728enc_G728dec	3.9	17-24	10-16	10-11
H263enc_MPGdec	4.1	12-19	11-15	9-12
MPGenc_GSMdec	4.7	8-18	9-10	1-6
H263enc_H263dec	4.8	25-32	16-26	13-18
H263dec_G728enc	5.0	20-23	11-21	0-14
H263enc_GSMenc	5.2	15-20	12-15	9-13
H263dec_G728dec	5.4	22-27	12-23	6-18
GSMenc_G728dec	5.8	29-31	12-27	1-14
MPGdec_GSMdec	5.9	15-17	12-15	2-12
H263dec_GSMenc	6.9	22-23	16-23	3-17
GSMenc_GSMenc	7.3	22-28	17-25	5-18
Average of min and max		17-23	11-18	5-13

(a)

Workload	IPC	A	B	C
MPGenc_MPGenc_MPGenc_MPGenc	5.3	26-31	15-29	14-18
MPGenc_MPGenc_G728dec_G728enc	6.6	26-31	13-30	12-16
H263dec_H263enc_GSMenc_H263enc	10.0	35-41	17-37	14-20
H263dec_H263enc_G728dec_H263dec	10.2	46-53	26-47	18-25
MPGdec_MPGenc_GSMenc_GSMenc	11.2	36-42	25-41	20-25
G728dec_H263dec_GSMdec_MPGdec	11.3	47-50	35-49	14-23
GSMenc_H263dec_GSMenc_H263dec	13.8	57-59	41-59	18-30
GSMenc_GSMenc_GSMenc_GSMenc	14.6	59-60	50-60	20-36
Average of min and max		42-46	28-44	16-24

(b)

Table 4. Range of % EPI savings of the best-energy CMP over the best-energy SMT for different performance regions with (a) two-thread workloads and (b) four-thread workloads. The workloads are ordered by the total IPC of the workload.

3.1.2 The Energy Efficiency of CMP vs. SMT

Overall results across all configurations.

Figures 2 and 3 and Tables 4(a) and (b) show that for all performance regions of all workloads, a CMP architecture gives the least EPI. For both two- and four-thread workloads, the difference between the best CMP and the best SMT EPIs is larger at the higher performance points. For the two-thread case, the difference becomes relatively low at the lower performance points. For the four-thread case, however, the absolute difference is much larger and stays large even at the lowest performance region.

More quantitatively, from the tables, we see that in the highest performance region (A), averaged across all workloads, the maximum difference between the EPI of the best CMP and the best SMT is 23% for two-thread and 46% for four-thread workloads. The average (across workloads) of the minimum difference in this region is 17% for two-thread and 42% for four-thread workloads. For the lowest performance region, these averages are a modest 13% and 5% for the two-thread case and a significant 24% and 16% for the four-thread case.

Implications for core architecture and frequency.

Best CMP core architecture

Figures 2 and 3 show that for all the workloads, the best CMP configuration uses a less or equally complex core (i.e., lower or same fetch/retire width) than the best SMT configuration at any given performance point. The total resources available to the CMP, however, are larger because the CMP has multiple such processor cores. This is the key reason why CMP consistently shows better EPI than SMT (discussed in more detail later).

For CMP, overall, the CMP-4 architecture is the best *in the regions where it has performance points.* For all such performance points, for the two thread case, the EPI of CMP-4 is better than or within 10% of the best CMP EPI for all but two workloads. For these latter two workloads, GSMenc_GSMenc and GSMenc_G728dec, the EPI of CMP-4 is 18% and 21% worse than the best CMP architecture, CMP-8 (However, the EPI is still less than the best SMT EPI). Similarly, for four-thread workloads, the CMP-4 configuration is again best overall and the EPI of CMP-4 is within 5% of the best CMP EPI for all but one workload (for regions where it has performance points). For the latter workload, four GSMenc threads, CMP-4 is 18% worse than the best CMP configuration, CMP-8.

More aggressive CMP architectures can provide much higher performance than CMP-4 at high frequencies. For example, for 4 of the two-thread workloads, CMP-8 can outperform CMP-4 at the highest frequency by 26% to 48%. For four-thread workloads, the performance difference is even more prominent. Similarly, at the lowest frequencies, less aggressive architectures can provide lower energy, albeit at lower performance (CMP-4 cannot get down to these performance points because of the bound on the minimum frequency).

Best SMT core architecture

For SMT, overall, the SMT-5 architecture is the best *in the regions where it has performance points.* Across

all such performance points and for all but one two-thread workloads, the EPI of SMT-5 is better than or at most 9% worse than the best SMT EPI. For GSMenc_GSMenc, it is up to 22% worse than SMT-8. For four-thread workloads, SMT-6 is the overall best (where it has performance points) and the EPI of SMT-6 is better than or at most 9% worse than the best SMT EPI. However, as with the CMP case, SMT-5 and SMT-6 do not exhibit the very highest and the very lowest performance points. At the highest frequency, SMT-8 outperforms SMT-5 by 20% to 36% for three two-thread workloads. For four-thread workloads, SMT-8 outperforms SMT-6 by 27% for one workload.

A case for adaptive architectures

The above data shows that it is possible to pick one core architecture for CMP and one core architecture for SMT to get close to the overall optimal EPI, if the regions of maximum or minimum performance are not the desired performance targets. A few of the workloads will see sub-optimal performance even in the middle performance range as described above. If the extreme regions are important performance targets or if the above workloads are important, then our data indicates that the best design would involve an adaptive processor that can change the active resources and fetch/retire width depending on the workload. Much recent research has been done on such processors [11, 2, 5, 15]. In particular, Hughes et al. describe a fairly straightforward algorithm to invoke the appropriate adaptation for multimedia applications, at the granularity of a full frame [11]. This algorithm could be easily modified for CMP and possibly modified for SMT processors as well.

The above discussion has focused on the appropriate choice of the core architecture. A similar analysis must also be done for the appropriate choice of the frequency. We find that the lowest EPIs are obtained across a range of frequencies for CMP and SMT, thereby supporting the use of DVS for these systems.

Finally, we note that CMP provides unique methods of adaptation that are not readily available to SMT architectures. Specifically, CMP can apply DVS and architectural adaptations independently to each core, depending on the type and amount of work to be done in each co-scheduled thread. We find that, 10 out of 14 two-thread workloads can save 9%-15% energy savings over the currently optimal CMP configuration, if independent DVS is available to each core.

Analysis.

We next analyze the underlying reasons for why CMP sees superior EPI to SMT.

Presence of an optimal core architecture.

The key to understanding our results is to see the relationship between the energy and complexity of the core architectures. Figures 4(a) and (b) plot the EPI for each core architecture for MPGenc_MPGdec running on two-thread systems and MPGenc_MPGdec_GSMenc_GSMenc running on four-thread systems respectively. This graph uses the performance point with SMT-2 at the highest frequency, since this is the highest performance point achieved by all architectures. We see that both CMP and SMT achieve a minimum EPI with moderately complex core architectures, with CMP using a slightly less complex core than SMT.

To understand why we see a minimum in the above curves, we use a relationship between EPI, power, and IPC derived as follows [11]:

$$EPI = \frac{Power \times Execution\_time}{Instruction\_count}$$

$$= C_{eff} V^2 f \frac{Execution\_time}{Instruction\_count},$$

where C_{eff} is the effective capacitance (i.e., product of capacitance and switching activity) [3]. For the purposes of this analysis, consider the voltage range where $f \propto V$ (i.e., where $V \gg V_{th}$). It follows that

$$EPI \propto C_{eff} \times \frac{Instruction\_count^2}{IPC^3} \times Execution\_time^2.$$

We note that for a specified voltage and frequency, for each architecture, C_{eff} is proportional to the (average) power for that architecture at that voltage and frequency. Previous work has shown that for the benchmarks studied, IPC is independent of frequency [9]. Thus, it follows that for a given performance for a given number of instructions, $EPI \propto \frac{P}{IPC^3}$.

Since IPC has a cubic effect on the EPI, having a higher IPC can quickly offset a moderate increase in average power. (Figure 5 shows how average power varies with processor core complexity for MPGenc_MPGenc, for 1-thread SMT (superscalar), 2-thread SMT and a 2-thread CMP.) When we increase the complexity of a core, initially we see a rapid decrease in EPI due to high IPC gains of more complex processors. However, after some point, we do not gain enough IPC (due to the limited parallelism exposed in the workload) to offset the increase in power. Therefore, we see an optimum architecture along the complexity axis. Note also that the above equation (and the earlier EPI vs. execution-time graphs) indicates that the optimum architecture is the same at all frequencies (for a given workload) in the range where $V \gg V_{th}$.

An example to show why CMP does better than SMT.

To understand why CMP does better than SMT, we consider an example scenario. Consider a thread T with IPC x and average power P on a superscalar (1-thread) processor. Assume that, corresponding to Figure 4, the superscalar processor has optimum EPI at complexity W, where W is the fetch/retire width of the processor. Now consider a workload with two copies of T and consider

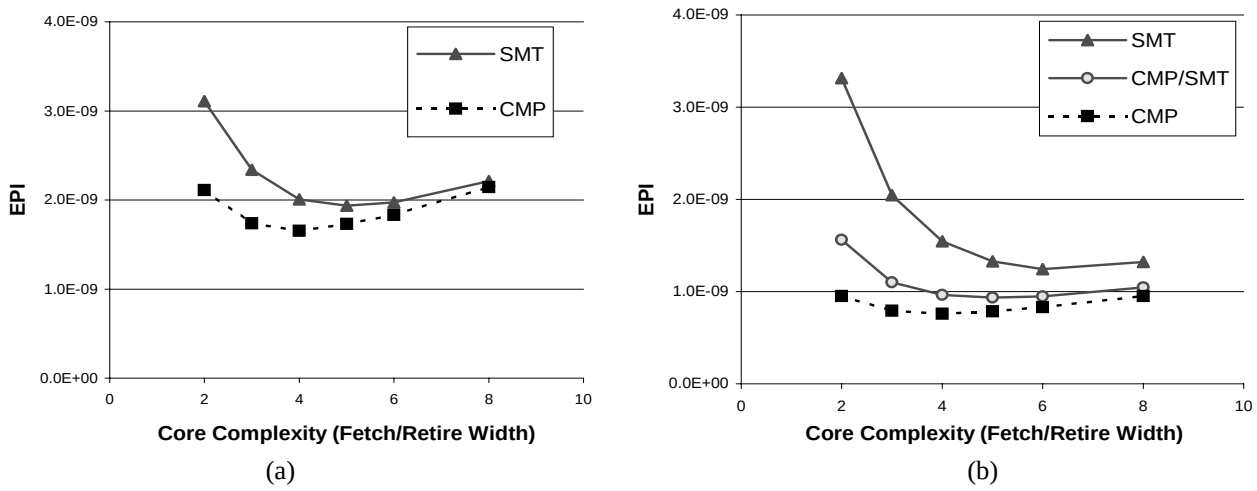

Figure 4. EPI for different processor cores at the performance point with SMT-2 at the highest frequency for (a) two-thread workload MPGenc_MPGdec and (b) for four-thread workload MPGenc_MPGdec_GSMenc_GSMenc, for SMT and CMP. Part (b) also shows data for a CMP/SMT architecture discussed later.

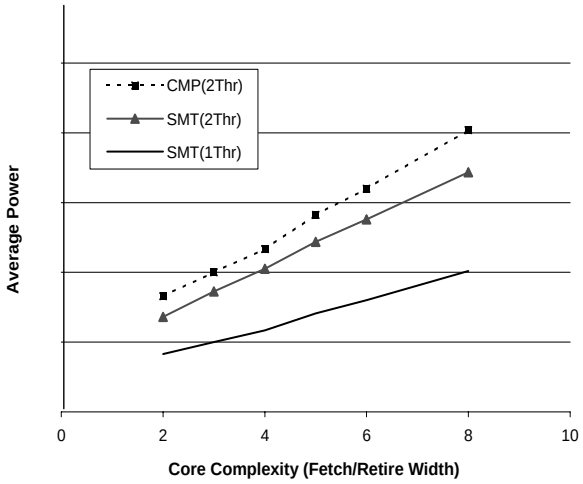

Figure 5. Average power for different processor cores at the highest frequency for a 1-thread SMT, 2-thread SMT, and 2-thread CMP.

that we want the same per-thread performance as the one-thread workload (i.e., IPC of $2x$). For CMP, this is achieved (roughly) with two processors with complexity W. The average power of this CMP is $2P$ and its EPI $\propto \frac{2P}{8x^3}$. Getting a better EPI for SMT requires that the SMT increase its IPC to $2x$ without consuming more than $2P$ of average power. From Figure 5, the average power of a 2-thread SMT with width W is smaller than $2P$. However, for the SMT to get an IPC of $2x$, it will likely need to increase the complexity of the core to some $W' > W$. (If the SMT could get $2x$ IPC with a core complexity of W, then it is generally unlikely that the superscalar with width W had the best EPI for the one-thread workload since many of its resources must have been idle). Therefore, SMT has to move up on the 2-thread average power curve (Figure 5) until it can get an IPC of $2x$. For our system, by the time the SMT reaches this point, its average power surpasses the average power of CMP. Note that increasing the frequency of the SMT does not help since that will only make the SMT climb a steeper curve and hit the CMP EPI faster.

Factors affecting the relative EPI of CMP and SMT.

On the basis of the example above, it follows that whether SMT or CMP has a lower EPI depends on the relative values of (1) the amount that SMT must move on its power curve to reach the optimal CMP IPC, and (2) the amount the SMT can move up its power curve before it exceeds the EPI for the CMP. The value for (1) depends on the symbiosis exploited by the applications and the IPC of the applications. The higher the symbiosis, the less the SMT must likely move, but the higher the IPC, the more it must likely move. Note that increasing number of threads supported contributes directly to high IPC (explaining why CMP did much better than SMT for 4-thread workloads). The value for (2) depends on how close the two-thread SMT

curve is to the CMP curve and on the steepness of the slope of the SMT curve. The closer the two curves and the steeper the SMT curve, the more likely that SMT will exceed the CMP EPI. Similarly, the more the SMT must move based on (1), the more likely it will hit the CMP EPI. Below, we elaborate on features that affect the distance between the SMT and CMP curves and the slope of the SMT curve, followed by an elaboration on the impact of symbiosis.

The distance between the SMT and CMP curves is significantly affected by the amount of clock gating. We assumed 10% of the resource power is dissipated even with clock gating. With perfect clock gating, the CMP curve will move closer to the SMT curve (in fact, all 3 curves will move down but the difference between the SMT and CMP curves will be minimal). This is because CMP will not be charged power for any underutilized resources and CMP has more underutilized resources than SMT, since CMP has more resources. If we did not have clock gating at all, all three curves will move up. Further, SMT will almost overlap with the superscalar curve, and the CMP curve will be twice as high as those curves. Thus, with more aggressive clock gating, the SMT curve moves closer to the CMP curve, making it difficult to outperform the CMP EPI. Conversely, without clock gating, SMT is likely to become more energy efficient.

The distance between the CMP and SMT power curves is also affected by symbiosis among the component applications of the workload. High symbiosis can move the SMT power curve up closer to the CMP (because it indicates there is likely to be higher resource utilization and hence switching activity per unit time).

The slope of the SMT curve is primarily determined by how much the power increases with complexity and hence depends on the power model. In our system, the slope is relatively steep, again working against the SMT.

From the above discussion, we see that the interaction between the symbiosis of the component applications in the multithreaded workload and the relative EPI of CMP and SMT is complex. High symbiosis helps SMT by requiring it to travel a shorter distance up the power curve to meet a given IPC. On the other hand, high symbiosis can hurt SMT by making the SMT curve too close to the CMP.

Finally, we note that the energy efficiency of CMP would be reduced if it could not increase its IPC linearly with the number of cores. This could happen due to bus contention or highly asymmetric workloads (since our cores are symmetric).

3.1.3 A Combined CMP/SMT Architecture

Our results so far show that CMP is by far the more energy efficient architecture than SMT for our workloads. However, SMT does have the advantage of potentially higher single-thread performance, which may be valuable for other workloads on the system and even in some cases for multimedia workloads. With this in mind, we propose an architecture that combines the single-thread performance advantage of an SMT with the energy efficiency advantage of a CMP. This combined CMP/SMT architecture consists of multiple SMT processor cores in a CMP configuration.

Workload	A	B	C
MPGenc_MPGenc_MPGenc_MPGenc	12-24	8-14	8-13
MPGenc_MPGenc_G728dec_G728enc	8-19	6-10	5-10
H263dec_H263enc_GSMenc_H263enc	15-28	5-18	2-7
H263dec_H263enc_G728dec_H263dec	13-27	3-15	0-6
MPGdec_MPGenc_GSMenc_GSMenc	17-27	13-18	9-14
G728dec_H263dec_GSMdec_MPGdec	9-14	0-9	-5-3
GSMenc_H263dec_GSMenc_H263dec	12-18	3-13	-5-6
GSMenc_GSMenc_GSMenc_GSMenc	15-22	14-19	-1-10
Average of min and max	13-22	7-15	4-9

Table 5. Range of % EPI savings of the best CMP over the best CMP/SMT for different performance regions.

As mentioned in Section 2, we study a CMP/SMT architecture with two processors, each of which is a two-thread SMT. We run the four-thread workload on this architecture using all the core architectures and frequencies. Figure 3 includes results for this architecture as well. Table 5 summarizes the magnitudes of the EPI improvements. Since CMP is the best architecture so far, we compare the EPI of the combined architecture with that of a CMP.

The results show that for all performance regions, the EPI advantage of CMP over CMP/SMT is much diminished, as compared to its advantage over the SMT architecture. Overall, CMP is only moderately better than the CMP/SMT architecture, in terms of EPI. Moreover, in performance region C, we see that sometimes the CMP/SMT architecture has lower EPI values than the CMP architecture. As seen from Figure 3, this happens at lower performance points where there is only one CMP configuration available in that performance region (due to a bound on the lowest frequency and the inability of CMP configurations to further increase execution time). The CMP/SMT configurations available in this region can further reduce voltage/frequency and achieve lower EPI values.

From Figure 3, we see that the core architecture for the least energy CMP/SMT is usually wider than that for the CMP at most performance points. This gives the CMP/SMT a potential performance advantage for single-thread workloads. Thus, CMP/SMT provides an interesting compromise point to achieve multithreaded EPI close to that of CMP with the single thread performance advantage of an SMT. Such an architecture also provides a more evolutionary migratory path to CMP for existing SMT processors.

3.2 Energy-Delay Product and Other Metrics

So far, we have focused on determining least-energy configurations for a given target performance. Here we discuss other energy related metrics.

Considering the lowest energy configurations overall (regardless of performance), we find that for CMP, this is always CMP-2 (except for 3 workloads where CMP-2 is up to 3% worse relative to the best EPI). This configuration is also 2%-18% better than the least energy SMT configuration for each workload and 0%-11% better than the least energy CMP/SMT configuration. For SMT, with two-thread workloads, SMT-3 is the lowest energy configuration overall (except for 3 workloads where it is 1% worse relative to the best EPI). Similarly, for CMP/SMT, CMP/SMT-3 is the lowest energy configuration overall (except for one workload where it is 1% worse).

To consider performance, the energy-delay product metric has been proposed in the past. The CMP with the overall best (i.e., lowest) energy-delay product is CMP-4. This is always the best except in a few workloads where it is at most 6% worse than the best configuration for that workload. For each workload, the best CMP configuration (in terms of energy-delay product) is 8% to 60% better than the best SMT configuration and 11% - 26% better than the best CMP/SMT configuration.

Thus, again we find that even in terms of the more conventional energy metrics, CMP is superior to SMT. However, looking at these metrics, we find that one CMP configuration mostly suffices, whereas the previous metric motivates more flexible and adaptive systems.

4 Conclusions

Several existing general purpose processors support multithreading either in the form of CMP or SMT. Multimedia applications, which are becoming a prominent workload and are inherently multithreaded, are a potentially good match for such systems. This paper evaluates the energy efficiency of general-purpose CMP and SMT procesors for multimedia workloads, with out-of-order processor cores. We explore the design space of CMP and SMT by simulating processor cores with different complexity. To evaluate energy efficiency, we develop a methodology by which we consider all configurations over a range of frequencies, allowing us to compare the energy of equal-performance configurations.

We evaluate two-thread and four thread multimedia workloads, derived from eight (sequential) multimedia benchmarks. We find that across the performance spectrum, CMP configurations are more energy efficient than SMT, for our systems and workloads. The relative difference in EPI between the best CMP and SMT generally increases with the total IPC of the workload and the number of threads supported. The difference is also larger for higher performance points. We analyze the reasons for these differences, identifying aggressive clock gating and the relatively steep power vs. core complexity curve for SMT as two reasons for our results.

We also found that at most performance points, the CMP with the 4-wide processor core architecture was optimal. There were, however, other performance points where other core architectures were optimal. This motivates the use of adaptive architectures that can deactivate parts of the core that lead to energy inefficiencies. Similar observations also motivate DVS. Further, we also found that exploiting heterogeneity in the CMP cores could further improve the CMP energy efficiency. SMT processors are not amenable to such adaptation.

Finally, even though CMP clearly gave the best EPI across the design space, it did so at lower complexity core configurations. Wider configurations of SMT can provide better single-thread performance. We propose and evaluate a hybrid CMP/SMT architecture, which is a CMP with SMT cores. We show that such an architecture is much better in terms of EPI for a four-thread system than a four-thread SMT and only moderately worse than a CMP architecture. Such an architecture could provide good energy efficieny if there is a need for wider cores to support single thread performance of very high IPC applications.

There are several directions for future work. We would like to study the effect of various real-time scheduling algorithms on these systems and also explore how adaptive cores can bring more energy savings.

References

[1] D. H. Albonesi. Selective Cache Ways: On-Demand Cache Resource Allocation. In *Proc. of the 32nd Annual Intl. Symp. on Microarchitecture*, 1999.

[2] R. I. Bahar and S. Manne. Power and Energy Reduction Via Pipeline Balancing. In *Proc. of the 28th Annual Intl. Symp. on Comp. Architecture*, 2001.

[3] L. Benini and G. D. Micheli. Dynamic Power Management Design Techniques and CAD Tools. In *Kluwer Acad. Publ.*, 1997.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proc. of the 27th Annual Intl. Symp. on Comp. Architecture*, 2000.

[5] A. Buyuktosunoglu et al. An Adaptive Issue Queue for Reduced Power at High Performance. In *Proc. of the Workshop on Power-Aware Computer Systems*, 2000.

[6] S. Egger, J. Emer, H. Levy, J. Lo, R. Stamm, and D. Tullsen. Simultaneous Multithreading: A Platfrom for Next-generation Processors. In *IEEE Micro, Sepetember/October*, 1997.

[7] R. Gonzalez and M. Horowitz. Energy Dissipation In General Purpose Microprocessors. In *IEEE Journal of Solid-state Circuits*, September 1996.

[8] L. Hammond, B. A. Nayfeh, and K. Olukotun. A Single-Chip Multiprocessor. In *IEEE Computer Special Issue on Billion-Transistor Processors*, September 1997.

[9] C. J. Hughes et al. Variability in the Execution of Multimedia Applications and Implications for Architecture. In *Proc. of the 28th Annual Intl. Symp. on Comp. Architecture*, 2001.

[10] C. J. Hughes, V. S. Pai, P. Ranganathan, and S. V. Adve. RSIM: Simulating Shared-Memory Multiprocessors with ILP Processors. *IEEE Computer*, February 2002.

[11] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications. In *Proc. of the 34th Annual Intl. Symp. on Microarchitecture*, 2001.

[12] Intel XScale Microarchitecture. http://developer.intel.com/design/intelxscale/benchmarks.htm.

[13] R. Jain, C. J. Hughes, and S. V. Adve. Soft Real-Time Scheduling on Simultaneous Multithreaded Processors. In *Proc. of the 23rd IEEE Int. Real-Time Systems Symp.*, 2002.

[14] S. Kaxiras, G. Narlikar, A. D. Berenbaum, and Z. Hu. Comparing Power Consumption of an SMT and a CMP DSP for Mobile Phone Workloads. In *Proc. of the Intl. Conf. on Compilers, Arch. and Synthesis for Embedded Systems*, 2001.

[15] R. Sasanka, C. J. Hughes, and S. V. Adve. Joint Local and Global Hardware Adaptations for Energy. In *Proc. of the 10th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2002.

[16] J. Seng, D. Tullsen, and G. Z. N. Cai. Power-Sensitive Multithreaded Architecture. In *Proc. of the 7th ASPLOS*, 1996.

[17] A. Snavely and D. Tullsen. Symbiotic Job Scheduling for a Simultaneous Multithreading Architecture. In *Proc. of the 9th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2000.

[18] D. Tullsen et al. Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor. In *Proc. of the 23th Annual Intl. Symp. on Comp. Architecture*, 1996.

[19] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A Power-Performance Simulator for Interconnection Networks. In *Proc. of the 35th MICRO*, November 2002.